

EAST Search History

| Ref # | Hits | Search Query | DBs | Default Operator | Plurals | Time Stamp |
|-------|-------|---|---|------------------|---------|------------------|
| S1 | 2 | "6074433".pn. | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2006/06/21 16:22 |
| S2 | 17 | ("4773007" "4807126" "5151991" "5293631" "5303357" "5396631" "5606697" "5640568" "5852734" "5901318" "5953531" "5958048" "5960171" "5987254" "6016397" "6026226").PN. OR ("6074433").URPN. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:27 |
| S3 | 439 | 717/151.ccls. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:28 |
| S4 | 119 | 717/152.ccls. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:28 |
| S5 | 123 | 717/155.ccls. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:28 |
| S6 | 171 | 717/156.ccls. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:29 |
| S7 | 79 | 717/157.ccls. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:29 |
| S8 | 280 | 717/159.ccls. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:29 |
| S9 | 123 | (optimiz\$5 or optimis\$5) same ((data adj (type or item)) or object or class) near5 (creat\$3 or instantiat\$3 or initializ\$5 or declar\$3 or defin\$3 or referenc\$3 or access\$3) and ("symbol table" or "constant pool") | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 17:14 |
| S10 | 72439 | memory near3 (manag\$4 or conservation or optimiz\$5 or optimis\$5 or reduc\$4 or minimal\$7) | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:44 |
| S11 | 48 | S9 and S10 | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:39 |
| S12 | 0 | ((data adj (type or item)) or object or class) near5 (creat\$3 or instantiat\$3 or initializ\$5 or declar\$3 or defin\$3 or referenc\$3 or access\$3) and ("not used" or "not referenced" or "not accessed") | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:45 |

EAST Search History

| | | | | | | |
|-----|------|--|------------------------------|----|-----|------------------|
| S13 | 5089 | (collaps\$3 or merg\$3 or consolidat\$3) near5 (hierarchical\$2 or nest\$3 or parent or child or hierarchy or tree or derived or subclassed or ancestor) | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:42 |
| S14 | 0 | S9 and S10 and S13 | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:43 |
| S15 | 355 | S10 and S13 | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:43 |
| S16 | 0 | memory near3 (manag\$4 or conservation or optimiz\$5 or optimis\$5 or reduc\$4 or minimal\$7) and (prun\$3 or reduc\$3 or eliminat\$3 or delet\$3) near3 undeclared | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 17:08 |
| S17 | 8020 | memory near3 (manag\$4 or conservation or optimiz\$5 or optimis\$5 or reduc\$4 or minimal\$7) and unused | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:44 |
| S18 | 9107 | ((data adj (type or item)) or object or class) near5 (creat\$3 or instantiat\$3 or initializ\$5 or declar\$3 or defin\$3 or referenc\$3 or access\$3) and ("not used" or "not referenced" or "not accessed" or unused or unitialized) | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:47 |
| S19 | 1901 | S17 and S18 | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:47 |
| S20 | 51 | S19 and S13 | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 16:47 |
| S21 | 2449 | memory near3 (manag\$4 or conserv\$5 or optimiz\$5 or optimis\$5 or reduc\$4 or minimal\$7 or reformat\$4 or restructur\$3) and (prun\$3 or reduc\$3 or eliminat\$3 or delet\$3) near3 (undeclared or unused or unnecessar\$3 or never) or ("not" adj (defined or instantiated or assing\$3 or created or declar\$3 or referenc\$3 or access\$3 or "point to")) | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 17:13 |
| S22 | 238 | S21 and S18 | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 17:13 |
| S23 | 220 | S22 not S20 | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 17:13 |

EAST Search History

| | | | | | | |
|-----|------|---|------------------------------|----|-----|------------------|
| S24 | 2804 | (optimiz\$5 or optimis\$5) same ((data adj (type or item)) or object or class) near5 (creat\$3 or instantiat\$3 or initializ\$5 or declar\$3 or defin\$3 or referenc\$3 or access\$3) | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 17:14 |
| S25 | 22 | S23 and S24 | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 17:47 |
| S26 | 62 | slicing near3 class | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/21 17:47 |
| S27 | 6 | slicing near3 class and (optimiz\$5 or optimiz\$5) | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/22 07:00 |
| S28 | 1 | "6301700".pn. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/22 07:01 |
| S29 | 6 | ("5161216" "5652899" "5778232" "5838979" "5854925" "5983020"). PN. OR ("6301700").URPN. | US-PGPUB; USPAT; USOCR | OR | OFF | 2006/06/22 07:02 |
| S30 | 12 | ("5983020").URPN. | USPAT | OR | OFF | 2006/06/22 07:39 |
| S31 | 200 | jax | USPAT | OR | OFF | 2006/06/22 07:39 |
| S32 | 148 | jax and (compil\$5 or optimiz\$5 or unused or remov\$3 or delet\$3 or merg\$3) | USPAT | OR | OFF | 2006/06/22 07:40 |
| S33 | 138 | jax and (compil\$5 or optimiz\$5 or unused or remov\$3 or delet\$3 or merg\$3) and (nam\$3 or renam\$3 or identif\$3 or siz\$3 or small\$3 or sizeof or "memory bytes") | USPAT | OR | OFF | 2006/06/22 07:52 |
| S34 | 9 | slic\$3 same renam\$3 | USPAT | OR | OFF | 2006/06/22 07:53 |


[Subscribe \(Full Service\)](#) [Register \(Limited Service\)](#)

Search: ☐ The ACM Digital Library ☐ The USPTO

+class +slicing

THE ACM DIGITAL LIBRARY

[Feedback](#) [Report a problem](#)

Terms used **class slicing**

Sort results by [Save results to a Binder](#) [Try an Advanced Search](#)
 Display results [Search Tips](#) [Try this search](#)
☐ Open results in a new window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Results

1 [Slicing class hierarchies in C++](#)

Frank Tip, Jong-Deok Choi, John Field, G. Ramalingam
 October 1996 **ACM SIGPLAN Notices , Proceedings of the 11th ACM conference on Object-oriented programming, systems, languages and applications OOPSLA '96**, Volume 31 Issue 10

Publisher: ACM Press

Full text available: [pdf\(2.08 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

This paper describes an algorithm for *slicing* class hierarchies in C++ programs. Given a C++ class hierarchy (a collection of C++ classes and inheritance relations) and a program P that uses the hierarchy, the algorithm eliminates from the hierarchy those members, member functions, classes, and inheritance relations that are unused in P , ensuring that the semantics of P is maintained. Class slicing is especially useful for generating a sliced program P is generated ...

2 [Slicing object-oriented java programs](#)

Zhenqiang Chen, Baowen Xu
 April 2001 **ACM SIGPLAN Notices**, Volume 36 Issue 4

Publisher: ACM Press

Full text available: [pdf\(828.91 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [terms](#)

This paper presents a new approach to represent dependence for object-c

software. The program dependence graph (PDG) consists of a set of PDG nodes, which are not connected. This new approach distinguishes data members for different objects. It represents the effects of polymorphism and dynamic bindings. Based on this, we introduce the concepts of partial slicing, class slicing and object slicing. In this algorithm, we slice not only statements but also data members ...


Keywords: class slicing, object slicing, partial slicing, program dependence


3 Slicing object-oriented software

Loren Larsen, Mary Jean Harrold

May 1996 **Proceedings of the 18th international conference on Software Engineering**

Publisher: IEEE Computer Society

Full text available:  [pdf\(1.26](#)

[MB\)](#) 

[Publisher](#)

[Site](#)

Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

Describes the construction of system dependence graphs for object-oriented programs, which efficient slicing algorithms can be applied. We construct these system dependence graphs for individual classes, groups of interacting classes and complete programs. For an incomplete system consisting of a single class or a number of classes, we construct a procedure dependence graph that simulates all possible execution paths in the class. For a complete system, we construct a system dependence graph ...

Keywords: Ada-95, C++ language, class libraries, class representation, efficient slicing algorithms, graphs, incomplete object-oriented programs, program construction, interacting classes, object-oriented programming, object-oriented slicing, procedure call simulation, procedure dependence graph, program methods, software libraries, software reusability, statically typed object-oriented subroutines, system dependence graphs

4 Analyzing information-flow in java program based on slicing technique

 Bixin Li

September 2002 **ACM SIGSOFT Software Engineering Notes**, Volume 32, Number 5

Publisher: ACM Press

Full text available:  [pdf\(632.50](#)


KB)Additional Information: [full citation](#), [abst](#)

Traditional information-flow analysis is mainly based on dataflow and c
In object-oriented program, because of pointer aliasing, inheritance, and
information-flow analysis become very complicated. Especially, it is dif
normal data and control-flow analysis techniques. some new approaches
analyze the information-flow between components in object-oriented pro
object-oriented program slicing technique is in ...

5 Inter-class def-use analysis with partial class representations

- ◆ Amie L. Souter, Lori L. Pollock, Dixie Hisley
September 1999 **ACM SIGSOFT Software Engineering Notes**, Proceed
ACM SIGPLAN-SIGSOFT workshop on Program ana
tools and engineering PASTE '99, Volume 24 Issue 5

Publisher: ACM Press


Full text available:  [pdf\(1.27 MB\)](#) Additional Information: [full citation](#), [abst](#)
[citings](#), [index term](#)

Object-oriented program design promotes the reuse of code not only thro
polymorphism, but also through building server classes which can be use
client classes. Research on static analysis of object-oriented software has
addressing the new features of classes, inheritance, polymorphism, and c
This paper demonstrates how exploiting the nature of object-oriented des
enable development of scalable static analyses. We p ...

6 Article abstracts with full text online: A brief survey of program slicing

- ◆ Baowen Xu, Ju Qian, Xiaofang Zhang, Zhongqiang Wu, Lin Chen
March 2005 **ACM SIGSOFT Software Engineering Notes**, Volume 30 1

Publisher: ACM Press

Full text available:  [pdf\(535.10 KB\)](#) Additional Information: [full citation](#), [abst](#)
[index terms](#)

Program slicing is a technique to extract program parts with respect to sc
computation. Since Weiser first proposed the notion of slicing in 1979, h
have been presented in this area. Tens of variants of slicing have been st
algorithms to compute them. Different notions of slicing have different p
different applications. These notions vary from Weiser's syntax-preservi
amorphous slicing which is not syntax-preserving, an ...


Keywords: debugging, dependence analysis, pointer analysis, program slicing

7 Untangling: a slice extraction refactoring

◆ Ran Ettinger, Mathieu Verbaere

March 2004 **Proceedings of the 3rd international conference on Aspect development**

Publisher: ACM Press

Full text available:  pdf(1.12 MB)

Additional Information: [full citation](#), [abstract](#)


Separation of concerns in existing code can be achieved by specific refactoring. Modern refactoring tools support a number of well-known refactoring transformations including method extraction. In this paper, we examine how method extraction is improved through program slicing. Furthermore, we show how a general *slice extraction* can be applied to untangle existing code by extracting aspects

8 Generalization of leaner object-oriented slicing

◆ Rob Law

January 1998 **ACM SIGSOFT Software Engineering Notes**, Volume 23

Publisher: ACM Press

Full text available:  pdf(177.00 KB)

Additional Information: [full citation](#), [abstract](#)


The Leaner Object-Oriented Slicing technique in [2], albeit effective in reducing code reduction from an object-oriented slice, fails to handle an object-oriented slice with multiple inheritance nets. That is, an object-oriented slice in a tree form with multiple branches is beyond the ability of leaner object-oriented slicing. This paper presents an approach which extends the applicability of code reduction by leaner object-oriented slicing. A generalized leaner object-oriented slice ...

9 Class hierarchy specialization

◆ Frank Tip, Peter F. Sweeney


October 1997 **ACM SIGPLAN Notices , Proceedings of the 12th ACM conference on Object-oriented programming, systems, languages and applications OOPSLA '97**, Volume 32 Issue 10

Publisher: ACM Press


Full text available:  [pdf\(2.29 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

Class libraries are generally designed with an emphasis on versatility and Applications that use a library typically exercise only part of the library's result, objects created by the application may contain unused members. An algorithm that *specializes* a class hierarchy with respect to its usage in a the algorithm analyzes the member access patterns for P 's variables, and classes for variable ...

10 Slicible rectangular graphs and their optimal floorplans

 October 2001 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 6 Issue 4


Publisher: ACM Press

Full text available:  [pdf\(267.25 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#), [reviews](#)

Rectangular dualization method of floorplanning usually involves topology followed by sizing. Slicable topologies are often preferred for their simplicity. While slicible topologies can be obtained efficiently, existing linear-time topology generation from a given rectangular graph does not guarantee success even if one exists. Moreover, the class of rectangular graphs, known as nonslicable graphs, do not have any slicible topologies ...


Keywords: Floorplanning, graph dualization, heuristic search, nonslicable graphs, slicible floorplans

11 Reengineering class hierarchies using concept analysis

 Gregor Snelting, Frank Tip

November 1998 **ACM SIGSOFT Software Engineering Notes , Proceedings of the ACM SIGSOFT international symposium on Foundations of software engineering SIGSOFT '98/FSE-6**, Volume 23 Issue 6

Publisher: ACM Press

Full text available:  [pdf\(1.31 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

The design of a class hierarchy may be imperfect. For example, a class C member m not accessed in any C -instance, an indication that m could be


moved into a derived class. Furthermore, different subsets of C 's members from different C -instances, indicating that it might be appropriate to split classes. We present a framework for detecting and remediating such design issues based on ...

12 Internet based audiovisual communication tools used in a joint engineering University of Alabama in Huntsville and the Ecole Supérieure et de Construction (ESTACA) in Paris, France

Robert L. Middleton, Robert A. Frederick

November 1998 **Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)**

Publisher: IEEE Computer Society

Full text available:  [html\(54.88 KB\)](#) Additional Information: [full citation](#), [abstract](#)

The University of Alabama in Huntsville, College of Engineering, in collaboration with the Colleges of Liberal Arts and Administrative Sciences, has for the last several years offered an annual senior level student design class using the Integrated Product Development approach. In year the participation in the class was expanded to include Engineering students from ESTACA in Paris, France. A small, but highly regarded, high school corporation in Linden, Alabama also participated in portions of the ...


Keywords: audiovisual, distant education, internet communication

13 Understanding class hierarchies using concept analysis

 Gregor Snelting, Frank Tip

May 2000 **ACM Transactions on Programming Languages and Systems**
Volume 22 Issue 3

Publisher: ACM Press

Full text available:  [pdf\(433.91 KB\)](#) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

A new method is presented for analyzing and reengineering class hierarchies. In this approach, a class hierarchy is processed along with a set of applications. A fine-grained analysis of the access and subtype relationships between objects and class members is performed. The result of this analysis is again a class hierarchy guaranteed to be behaviorally equivalent to the original hierarchy, but in which only the members that are required are retained.

Keywords: class hierarchy reengineering, concept analysis

14 Technical correspondence: Slicing Z specifications

◆ Fangjun Wu, Tong Yi

August 2004 **ACM SIGPLAN Notices**, Volume 39 Issue 8

Publisher: ACM Press

Full text available:  [pdf\(285.76 KB\)](#) Additional Information: [full citation](#), [abstract](#)

Program slicing is a well-known technique that has been broadly applied in software engineering areas, such as understanding, debugging, testing and program slicing. Programs written in a high-level language have been widely studied, but very little work is involved in slicing formal specifications. In this paper, we present a method of specification slicing. First, we analyze dependences in Z and identify a kind of dependence, i.e. logic dependence. See ...

Keywords: dependence analysis, formal specification language, program specification slicing

15 Automated abstraction of class diagrams

◆ Alexander Egyed

October 2002 **ACM Transactions on Software Engineering and Methodology**, Volume 11 Issue 4

Publisher: ACM Press

Full text available:  [pdf\(1.76 MB\)](#) Additional Information: [full citation](#), [abstract](#), [index terms](#)

Designers can easily become overwhelmed with details when dealing with class diagrams. This article presents an approach for automated abstraction that allows to "zoom out" on class diagrams to investigate and reason about their big picture. The approach is based on a large number of abstraction rules that individually are powerful, but when used together, can abstract complex class structures. The paper presents those abstraction rules and an algorithm for ...

Keywords: Class abstraction, class diagrams, class patterns, reverse engineering, transformation, unified modeling language

16 Analysis of Several Task-Scheduling Algorithms for a Model of Multiprog
Systems



K. L. Krause, V. Y. Shen, H. D. Schwetman

October 1975 **Journal of the ACM (JACM)**, Volume 22 Issue 4

Publisher: ACM Press

Full text available: [pdf\(1.76 MB\)](#) Additional Information: [full citation](#), [reference terms](#)

17 Equivalence analysis and its application in improving the efficiency of pro:



Donglin Liang, Mary Jean Harrold

July 2002 **ACM Transactions on Software Engineering and Methodolo**
Volume 11 Issue 3

Publisher: ACM Press

Full text available: [pdf\(457.78 KB\)](#) Additional Information: [full citation](#), [abstract terms](#)

Existing methods for handling pointer variables during dataflow analysis: analyses inefficient in both time and space because the data-flow analysis propagate large sets of data facts that are introduced by dereferences of p This article presents *equivalence analysis*, a general technique to improv data-flow analyses in the presence of pointer variables. The technique id relations among the memory locations ...

Keywords: Alias analysis, data-flow analysis, program slicing

18 Computational sample complexity



Scott Decatur, Oded Goldreich, Dana Ron


July 1997 **Proceedings of the tenth annual conference on Computation:**

Publisher: ACM Press

Full text available: [pdf\(2.12 MB\)](#) Additional Information: [full citation](#), [reference terms](#)

19 Concurrent garbage collection using program slices on multithreaded processes


◆ Manoj Plakal, Charles N. Fischer

October 2000 **ACM SIGPLAN Notices**, **Proceedings of the 2nd international conference on Memory management ISMM '00**, Volume 36 Issue 1**Publisher:** ACM PressFull text available:  [pdf\(957.62 KB\)](#) Additional Information: [full citation](#), [abstract terms](#)

We investigate reference counting in the context of a multi-threaded architecture exploiting two observations: (1) reference-counting can be performed by program slice of the mutator that isolates heap references, and (2) hardware that microprocessors in the near future will be able to execute multiple contexts on a single chip. We generate a reference-counting collector as a transformer on an application and then execute this slice in ...

20 System-dependence-graph-based slicing of programs with arbitrary interprocedural flow




Saurabh Sinha, Mary Jean Harrold, Gregg Rothermel

May 1999 **Proceedings of the 21st international conference on Software Engineering****Publisher:** IEEE Computer Society PressFull text available:  [pdf\(1.31 MB\)](#) Additional Information: [full citation](#), [references](#), [index terms](#)**Keywords:** interprocedural slicing, program slicing, system dependence

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#)

The ACM Portal is published by the Association for Computing Machinery
ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)